10/581862

IAP2 Rec'd PCT/PTO 05 JUN 2006

DaimlerChrysler AG

Loading of software modules

The invention relates to a method for loading a software module into a processor unit in a controller in a means of transport, the software module being executable in a plurality of controllers and the controllers interchanging data via a data bus.

DE 196 31 309 A1 discloses a microprocessor arrangement for a vehicle control system having a plurality of microprocessor systems which are connected to one another by bus systems.

It is the object of the present invention to optimize the processor utilization level in controllers which are networked to one another.

The invention achieves this object by means of the features of claim 1. Accordingly, a selection is made regarding the controller on which the software module is loaded, on the basis of the computation capacity of the controllers which are currently in operation. The selection method ensures that the software module currently has sufficient computation capacity available on the loaded controller for executing its processes and is not started on a controller on which there is currently insufficient computation capacity. The selection method allows targeted utilization of free computation capacities in a complex of controllers which can communicate with one another.

Preferably, the computation capacity of the controllers is ascertained in rotation or upon request. This has the advantage that it is known which controller currently has how much free computation capacity. This information can accordingly be used to control the

loading of the software module onto a particular controller. The free computation capacity of a controller is dependent on the tasks which are currently to be handled by this controller. This is therefore subject to fluctuations and needs to be communicated to the other controllers.

Advantageously, the computation capacity of a controller is ascertained from the processor utilization level and the processor type, so that there is the assurance that even with different processor types the free computation capacity is determined correctly, in particular not only the processor utilization level is used.

Preferably, the software module is started on the controller with the maximum free computation capacity, so that controllers with relatively little computation capacity are not burdened with executing the software module.

Preferably, the controller on which the software module is running compares its computation capacity with the computation capacity of the other controllers. On the basis of the comparison, the software module is terminated or continued by the controller. This has the advantage that the software module can be turned off in the event of processor utilization level alterations on the controller.

Advantageously, termination of the software module prompts ascertainment of which of the other controllers provides the maximum free computation capacity. In addition, the software module is started on this controller.

Advantageously, the software module is executable on the controllers, because otherwise the software module

cannot be loaded by the controllers. In addition, the controllers are in ongoing operation. The software module is therefore loaded at the runtime of at least the operating system and possibly of further software modules which have been loaded on the controller in question.

Preferably, the software module sends an identifier about its operating state and its operating controller, that is to say an identifier for the controller on which the software module is running, to the data bus in rotation or upon request. This ensures that the correct operation can be checked and the software module can be influenced directly.

There are now various options for advantageously refining and developing the disclosure of the present invention. In this regard, reference is made firstly to the subordinate claims and secondly to the explanation of an embodiment below. It is also necessary to include the advantageous refinements which are obtained from any combination of the subclaims. The drawings show an embodiment of the inventive method and an apparatus, and, in each case in a schematic illustration,

figure 1 shows an apparatus for carrying out the inventive method, and

figure 2 shows a method sequence for carrying out the inventive method.

An overview of an apparatus for carrying out the inventive method as shown in figure 1. The components involved in a bus system in a means of transport 9 are connected to one another by means of a data bus 8. The components involved preferably comprise controllers, sensors and actuators. In the schematic figure 1, the

components are controllers 1, 3, 5 with appropriate software modules 2, 4, 6, 7 running thereon.

The operating systems used allow the controllers 1, 3, 5 or their software modules 2, 4, 6, 7 to communicate with one another. In this case, standards are used which are already established in the field of the software for vehicles. Some of these standards are OSEK - open systems and their interfaces for electronics in motor vehicles - adopted into ISO 15765-2 (http://www./osek-vdx.org), as a transport protocol between controllers, or the Keyword Protocol 2000, adopted into ISO 14230 (http://www.iso.org), for transmitting diagnostic data and providing diagnostic services.

The communication protocol available is the Keyword Protocol 2000 (KWP 2000), which is used in the vehicle industry as a communication protocol for diagnostic services and meets ISO 14230. Any other communication protocol may be used, however, provided that it performs the tasks below or meets ISO-14230.

The controllers 1, 3, 5 have at least one microcontroller with a processor, memory and input/output unit for performing the controller function, a communication controller for implementing the communication protocol and a transmission/reception unit for connecting to the data bus 8. The data bus 8 is in the form of a CAN data bus with appropriate protocol functionality.

The software modules 2, 4, 6, 7 correspond to software-controlled applications which run on the respective controller 1, 3, 5. The controllers 1, 3, 5 are able to load a plurality of software modules.

The controllers 1; 3; 5 load the software modules 2; 4; 6 stored in the microcontroller's memory into their processor unit. These software modules 2; 4; 6 perform the primary tasks of the relevant controller 1; 3; 5. The software module 7 may additionally be loaded by the controllers 1, 3, 5. The software module 7 corresponds to a secondary task of the controllers 1, 3, 5. The software module 7 is likewise stored in the memory of the microcontroller in the controllers 1, 3, 5.

By way of example, the controller 1 uses the software module 2 to undertake engine control as the primary task, the controller 3 uses the software module 4 to undertake power train control as the primary task, and the controller 5 uses the software module 6 to undertake control of the braking system as the primary task.

As the secondary task, the software module 7 undertakes the calculation and creation of diagnostic data, for example, which are suitable for display in the means of transport 9 and/or storage at a central location in the means of transport 9.

The software module 7 may be started in any controller 1, 3, 5. To this end, the controllers 1, 3, 5 support the input/output demands of the software module 7.

By way of example, operating data from sensors or actuators on the data bus 8, such as oil temperature, servomotor position etc. are forwarded from the respective controller 1, 3, 5 as data to the software module 7.

The process time required by the software module 7 corresponds to the total time in which the software module 7 used a particular processor from when it was started to the execution of its task. The processor

time is particularly dependent on the clock frequency of the processor type used in the microcontroller of a controller 1, 3, 5.

The controllers 1, 3, 5 operate in process cycles, i.e. after a particular time has elapsed a process cycle needs to be terminated and the data ascertained in the process need to be output onto the data bus 8. The process cycle then starts again. The process cycle for the controllers 1; 3; 5 is determined by the software modules 2; 4; 6 of the primary task and/or the operating system and/or the bus protocol. Accordingly, the processes which arise from the software modules 2, 4, 6 running on the processor of the respective microcontroller in the controller 1, 3, 5 are called primary processes.

When a process cycle or a process cycle time has elapsed, the controllers 1, 3, 5 send data to the data bus 8 which characterize their current processor utilization level and processor type used. From these data, the controllers 1, 3, 5 can ascertain the utilization level of the other controllers 1, 3, 5.

The utilization level of a processor as a result of handling the primary task of a controller 1, 3, 5 is not uniform. The processor's utilization level varies depending on the demand from the primary task. By way of example, the processor utilization level in the controller 5 as a result of the primary process is higher when braking than when not braking. Similarly, the processor utilization level of the controller 3 is higher when changing gear than when not changing gear.

The software module 7 can run on the various controllers 1, 3, 5. The decision regarding on which of the controllers 1, 3, 5 the software module 7 is started is dependent on the computation capacity, that

is to say the processor utilization level and the processor type, of the respective controller 1, 3, 5.

The inventive method will now be explained with reference to the flowchart shown in figure 2, it subsequently being assumed that the processors in the controllers 1, 3, 5 are of identical type, that is to say particularly have the same clock frequency, and that the controllers 1, 3, 5 are in ongoing operation:

Check 10:
A check is performed to determine whether and on which controller 1, 3, 5 the software module 7 is running. This check needs to be performed in rotation, that is to say in particular time periods, since each of the controllers 1, 3, 5 is able to turn off the software module 7 when processor utilization level is high. As soon as the software module 7 has been turned off, the software module 7 needs to be started again. The check to determine whether and on which controller 1, 3, 5 the software module 7 is running is performed by virtue of the software module 7 sending an appropriate identifier which contains these data to the data bus 8 in rotation or upon request.

Decision 20:

By way of example, an appropriate identifier for the software module 7 was not able to be ascertained on the data bus 8 in step 10, which means that it is necessary to branch to step 30.

Computation capacity 30:
For this, it is established which of the controllers 1, 3, 5 involved in the data bus 8 has the maximum free computation capacity, that is to say the lowest processor utilization level in relation to the processor clock frequency. This information can be

obtained by virtue of the controllers 1, 3, 5 involved sending in rotation or by means of a request. By way of example, the controller 3 currently needs to have the maximum free computation capacity.

Start software module 40:
The software module 7 is started by the controller 3 determined in the previous step 30.

Software module running 50:
As soon as the software module 7 has been started correctly, it sends an identifier about its operation state and its operating controller, that is to say the controller on which the software module 7 is running, to the data bus 8 in rotation or upon request.

Check 10:
The check in rotation ascertains whether and, if appropriate, which identifier for the software module 7 is present on the data bus.

Decision 20:
Since step 10 shows that the software module 7 is running on controller 3, it is necessary to branch to step 60.

Decision 60:
The controller 3 ascertains its own current processor utilization level within a process cycle and compares it with the current computation capacity of the other controllers 1, 2 within a process cycle. To this end, it either requests the information regarding computation capacity, that is to say processor utilization level and processor type, from the controllers 1, 2, or the controllers send this information to the data bus 8 in rotation.

If the utilization level of the processor in the controller 3 is lower in comparison with the utilization level of the processors in the other controllers 1, 2, no action occurs. The software module 7 continues to run on the controller 3. The branch to checking step 10 is effected in rotation.

If the utilization level of the processor in the controller 3 is higher in comparison with the utilization level of the processors in the other controllers 1, 2, the branch to step 70 is effected.

Turn off software module 70:
The software module 7 in the controller 3 is turned off. In addition, the controller 3 uses its data to ascertain the controller 1, 2 with the currently maximum free computation capacity. This will be the controller 1 by way of example.

Start software module 40:
The software module 7 is started by the controller 1 determined in the previous step 70.

Software module running 50:
As soon as the software module 7 has been started correctly, it sends an identifier indicating that and on which controller it is running to the data bus 8 in rotation or upon request.

It is also possible for a plurality of different software modules to be distributed over the controllers 1, 3, 5 as secondary tasks. In addition, the controllers 1, 3, 5 may also perform a plurality of primary tasks.

The inventive method is preferably implemented at the operating system level of the controllers 1, 3, 5.

The data bus 8 may also be provided, by way of example, in the form of a FlexRay bus, in the form of an optical MOST or D2B bus or in the form of an electrical LIN bus in a means of transport, particularly a vehicle.

Advantageously, the inventive method may also be used in safety-related systems in vehicles. To increase failsafety, these systems are of redundant design, so that if a controller fails, for example, it is possible to change over to a controller of redundant design. Hence, systems of redundant design contain a plurality of controllers of the same type on which the same primary process runs, namely the software application of redundant design. The necessary similarity of the controllers of redundant design implies that a software module which can be executed on one of these controllers of redundant design can also be executed on the associated other controllers belonging to the system of redundant design. This may be used for the application of the inventive method by virtue of coordinate software applications additionally running on a controller in the redundant system.

In the method described hitherto, the processor power of the controllers 1, 3, 5 is in a form such that the software module 7 for the respective primary task of the controller 1, 3, 5 can always be connected in without the primary process having to dispense with process time. This primary process therefore always receives priority over all other processes which are running on the processor. Should this not be the case, it is additionally necessary to check in step 60 and in step 30 whether the free computation capacity available on the respective controller 1, 3, 5 is sufficient for handling the secondary task. Should this not be the case, the software module 7 cannot be started in the relevant controller. For this calculation, the controllers 1, 3, 5 require advance knowledge of the

process time for the software module 7 for a particular process type.

The inventive method may likewise be applied if the processor types in the controllers 1, 3, 5 are different. When the free computation capacity is determined, it is then necessary to take account not only of the processor utilization level but also of the processor type, that is to say particularly of the processor clock frequency.

The inventive method can also be extended to controllers whose microcontrollers have a plurality of processors.

The inventive method may also be controlled by means of a central controller. This has the advantage that the central controller can distribute the appropriate software application to the controller in step 40 in addition to the decision and computation steps 20, 60, 30.